THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

The Stochastic Loss of Spikes in Spiking Neural P Systems: Design and Implementation of Reliable Arithmetic Circuits

Citation for published version:
Xu, Z, Cavaliere, M, An, P, Vrudhula, S & Cao, Y 2014, 'The Stochastic Loss of Spikes in Spiking Neural P Systems: Design and Implementation of Reliable Arithmetic Circuits', *Fundamenta Informaticae*, vol. 134, no. 1-2, pp. 183-200.

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Fundamenta Informaticae

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 06. May. 2023

The Stochastic Loss of Spikes in Spiking Neural P Systems: Design and Implementation of Reliable Arithmetic Circuits

Zihan Xu[1], Matteo Cavaliere[2], Pei An[1], Sarma Vrudhula[3], Yu Cao[1]

[1] School of Electrical, Computer and Energy Engineering, Arizona State University, USA
[2] School of Informatics, University of Edinburgh, UK
[3] School of Computing, Informatics and Decision Systems Engineering, Arizona State University, USA

Summary. Spiking neural P systems (in short, SN P systems) have been introduced as computing devices inspired by the structure and functioning of neural cells. The presence of unreliable components in SN P systems can be considered in many different aspects. In this paper we focus on two types of unreliability: the stochastic delays of the spiking rules and the stochastic loss of spikes. We propose the implementation of elementary SN P systems with DRAM-based CMOS circuits that are able to cope with these two forms of unreliability in an efficient way. The constructed bio-inspired circuits can be used to encode basic arithmetic modules.

1 Reliability and SN P Systems

Membrane computing (known also as P systems) is a model of computation inspired by the structure and the functioning of living cells, [8]. A P system is a parallel computing device based on multiset rewriting in compartments where a global clock is assumed and each rule of the system is executed in one time step. Spiking neural P systems (in short, SN P systems) have been introduced as computing devices inspired by the structure and functioning of neural cells (a friendly introduction to this model of computation is given in [7] while a more complete review is provided in the corresponding chapter of the handbook, [8]).

The main idea of an SN P system is to have several one-membrane cells (called neurons) which can hold any number of spikes; each neuron fires (spikes) in specified conditions (after accumulating a specified number of spikes). In the basic

Corresponding author: Yu.Cao@asu.edu

definition of SN P systems, [7], the functioning of the system is synchronous: a global clock is assumed and, in each time unit, each neuron that can use a rule does it. The system is synchronized but the work of the system is sequential: only (at most) one rule is used in each neuron. In the basic model, one of the neurons is considered to be the output neuron and its spikes are also sent to the environment. A possibility is that steps when (at least) one spike is emitted by the output neuron are marked with $1s$, while the other steps are marked with 0s. The binary sequence obtained in this manner is called the *spike train* of the system.

To a spike train one can associate various numbers, which can be considered as computed by an SN P system. Synchronized SN P systems, with spiking rules in the standard form (i.e., they produce only one spike) are universal, they can characterize the family of Turing computable sets of natural numbers, [7].

In many computational results obtained in the area the synchronization between different processes plays an important role (see a discussion in the chapter dedicated to the role of time in membrane systems, [8]). However implementing synchronization is not always easy or even possible, often even similar processes can take distinct and unpredictable times to be completed.

For these reasons, starting from the basic idea that different chemical reactions may take different times to be executed (or to be started, when enabled) a timed model of P system was introduced, [3], where to each rule of the system is associated a time of execution. The goal was to understand how time could be used to influence the result produced by the P system and, possibly, how to implement powerful time-free systems where the output produced was independent of the timings associated to the rules. Time-free systems have been extensively studied in the context of membrane systems and extended to spiking neural P systems, [2, 14], including to the solution of hard computational problems, [13]. In all these works the underlying basic issue is to understand how a computation can be encoded in a system composed of unreliable components. However, the presence of unreliable components in SN P systems can be considered in many different aspects (e.g., in the form of a stochastic delays of the spiking rules, as proposed in [2], or the stochastic loss of spikes, that we introduce in this paper). Aside from challenging theoretical problems, [3], the presence of unreliable components pose an important constrains on the possible hardware implementations of SN P systems. Here we focus on the reliability problems in the context of engineering of basic circuits, inspired by the concepts and the functioning of SN P systems, using DRAM-based CMOS circuits, designed in a way to cope with the random loss of spikes and the stochastic application of rules.

2 Neurons and Arithmetic Circuits

2.1 Two Basic Types of Neurons

In this paper we do not use the complete definition of neurons as defined in the SN P systems literature but we restrict our attention to very specific types of neurons

that are the constituents of the presented hardware implementation. Moreover we assume the reader familiar with the basic concepts of SN P systems, [7, 2], and with the notions and terminology from the area of DRAM-based CMOS circuits.

Essentially, SN P systems computations are based on a combination of firing rules and forgetting rules [7, 2]. Firing rules allow a neuron to emit and broadcast spikes to other neurons. Inspired by this functioning, we consider the simple possibility to determine the applicability of a firing rule by checking the total number of input spikes (spikes received) against a given threshold. To mimic a biological neuron, when a neuron sends out spikes, becomes inactive for a specified period of time, i.e., the refractory period, during which the neuron does not accept new input and cannot fire. When the neuron is firing, there is also a delay associated between the input and output spikes. In the general framework, [7, 2], a set of firing rules, with different firing expression and numbers of output spikes, can be associated to a neuron. If no firing rule can be applied, there is the possibility to apply a forgetting rule, which removes a pre-defined number of spikes from the neuron, without firing any output spikes. Figure 1 illustrates one example that abstracts a biological neuron to a digital-like SN P neuron, as those that we consider in this paper: a stands for the spike (the signal sent); λ means that there is no output spike; and the natural numbers represent the number of spikes. Input enters the neuron sequentially (each vertical bar corresponds to an entering spike). At the end of either the firing or the forgetting operation, all input spikes have been consumed.

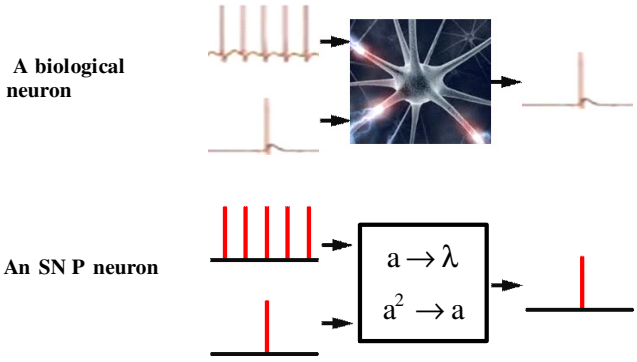

Fig. 1. An example of an SN P neuron that fires a spike for every two input spikes; it does not fire if there is only one input spike. Inputs (spikes) arrive sequentially with a prescribed temporal order.

A general SN P neuron may contain several rules, [7]. In this paper, because of the restrictions on the proposed hardware implementation, two basic types of SN P neurons are introduced and used, shown in Figure 2: high-pass (HP) neuron and low-pass (LP) neuron, defined with complementary spiking rules (we also call

them firing rules). They can receive an unlimited number of inputs and produce only one output. The threshold in their firing rules is set to be 2. This means that HP neurons fire when receive two or more than two input spikes, while LP neurons fire when there is only one input spike. In this case, because of the specific type of neurons, we prefer to use an explicit (threshold-like) syntax (instead of the usual and more general regular expression, [7]).

Fig. 2. The definition of two basic neurons. Notice the characteristic shape that identifies each type, as we are using them in the successive constructions.

During the hardware implementation of an SN P system, the role of these simple neurons is similar to that of a 2-input NAND gates in the Boolean logic: they may not provide the most compact design but they are sufficient to be the basis of more generic implementations. In fact, HP and LP neurons could be used to build more complicated neurons in the following manner. A neuron with several rules can be seen as operating in two consecutive steps: counting the input spikes received and then deciding which rule to apply (we will call these two stages as the counting and decision steps). In order to implement a more complex neuron using HP and LP neurons, we can start by considering these two steps separately. The counting operation sums up all the input spikes and the decision operation selects the rule to be applied based on the number of input spikes. For example, Figure 3 presents a simple counter using HP and LP neurons, which can count up to 3 spikes. The idea of the proposed design is to convert the input spikes into binary code, which also functions as a 1-bit adder. As shown in Figure 3, the 1-bit adder has three inputs (In_1, In_2 and In_3) and two outputs (O and E). Output O (odd) will produce a spike if there are one or three input spikes; on the other hand, output E (even) will produce a spike if there are two or more input spikes. Hence E represents the most significant bit (MSB) while O represents the least significant bit (LSB).

The 1-bit adder functions in the following way. If there is only one input spike, neuron 3 will apply the forgetting rule and will not fire; thus neuron 5 will also not fire; neurons 1 and 2 will pass the input spike to neuron 4 and neuron 4 will emit a spike. If there are two input spikes, neuron 3 will execute the firing rule emitting a spike that will be passed to E by neuron 5; neuron 4 will get two input

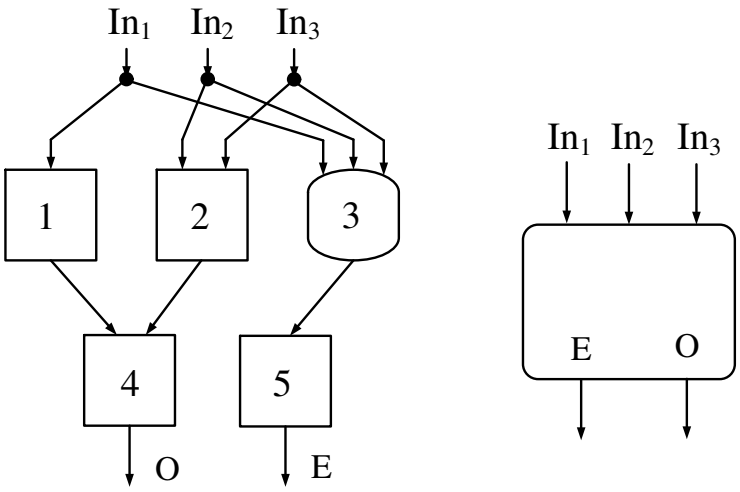

Fig. 3. Schema and corresponding symbol of a 1-bit adder. This 1-bit adder has three parallel inputs and converts the number of inputs into binary code.

spikes (one spike from In_1 and one spike from In_2 or In_3) or no input spikes (two input spikes from In_2 and In_3) and thus no spike will be emitted from O. If there are three input spikes, neuron 3 will execute the firing rule emitting a spike which will be passed to E by neuron 5; neuron 1 will pass the input spike to neuron 4 and neuron 2 will execute the forgetting rule, thus neuron 4 will see only one spike and then fire a spike out. In conclusion, from the described dynamics, it is possible to see that the E output works as the carryout and the O output works as the sum. The presented 1-bit adder converts the number of input spikes (up to 3) into binary representation. With an expanded adder, an arbitrary number of input spikes can be converted into binary representation. Figure 4 presents a converter, which can sum up to 12 input spikes. With similar circuits we can perform the counting of the received input spikes in more complex neurons.

On the other hand, the decision steps (implemented by a "ruler" neuron) works like a look up table. Different inputs $a_0a_1 \cdots a_n$ will activate different parts which correspond to different rules (see Figure 5). These parts can be designed separately and then combined together. For example, the circuit corresponding to the rule $a^7 \to a$ is shown in Figure 5. The deactivate signal comes from the output of the rulers with a larger threshold. All the firing rulers are connected to an LP neuron to assemble the final output. The forgetting ruler is similar to the firing ruler. The only difference is that it is not connected to the output neuron.

Figure 6 presents an instance of a neuron that can be implemented by an opportune combination of HP and LP neurons. We will briefly explain how such complex neuron can be engineered. Since the firing rules in the target neuron have

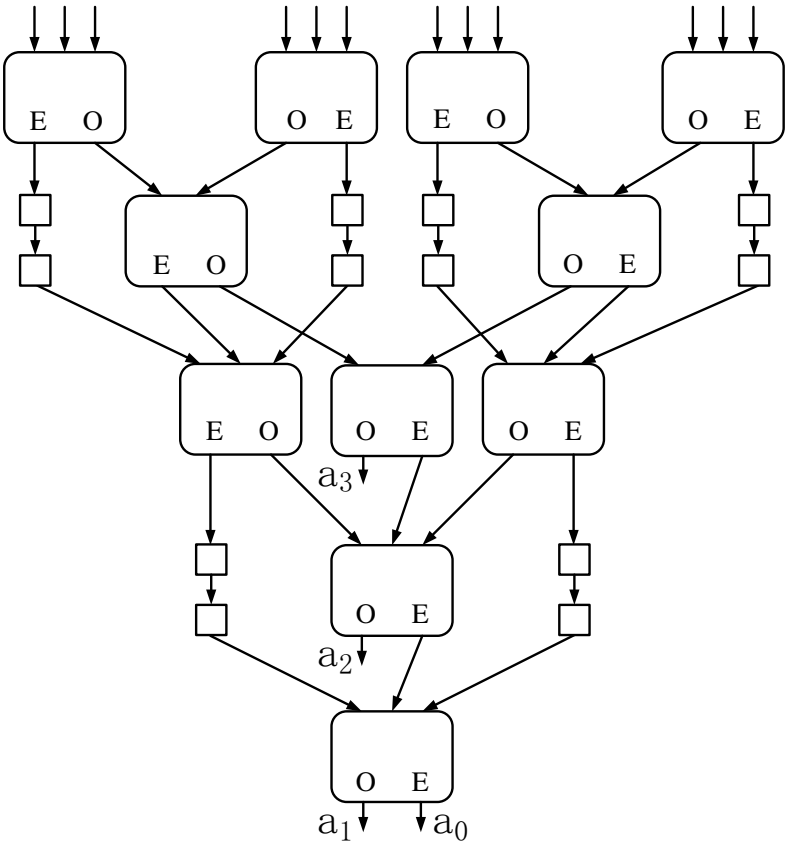

Fig. 4. Design of a multi-bit adder.

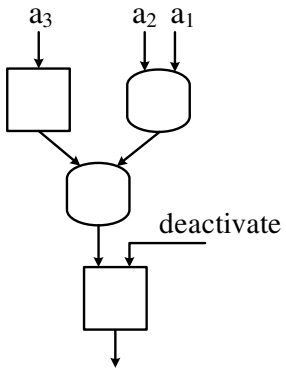

Fig. 5. Ruler design of rule $a^7 \rightarrow a$.

a threshold of 6 spikes, the implementation needs a 6-spikes counter shown in Figure 7.

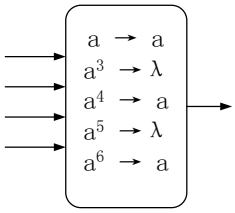

Fig. 6. Instance of a more complex neuron that can be implemented by a combination of HP and LP neurons.

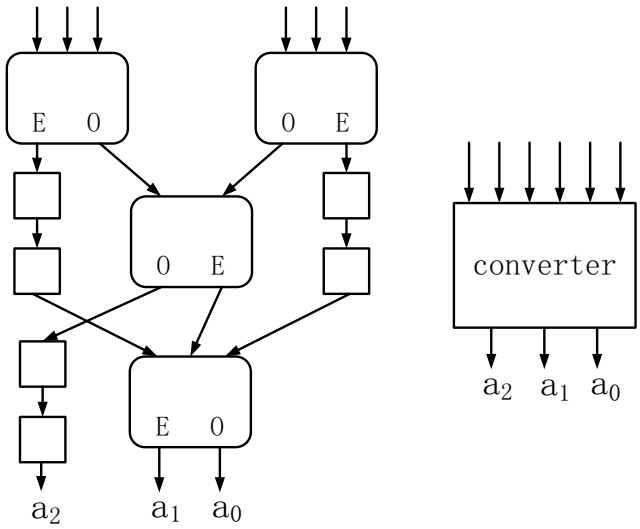

Fig. 7. Design (left) and symbol (right) of a 6-spike counter.

Figure 8 presents the design of the neuron. Each sub-part (in dashed rectangles) corresponds to one rule in the target neuron. Notice that a^2 does not lead to any firing or forgetting rule and it is used only to generate a feedback to the neuron.

After combining the converter and the ruler described in Figure 7 and Figure 8, the implemented neuron is shown in Figure 9. The presented design procedure can be applied to any number of inputs and any combination of rules. In an informal way this shows how the basic HP and LP neurons can be used to engineer more complex neurons.

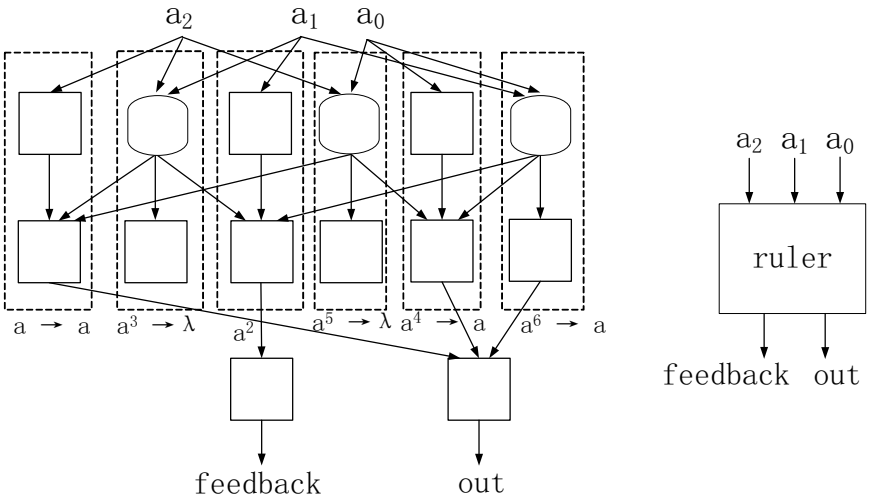

Fig. 8. Design (left) and symbol (right) of the ruler.

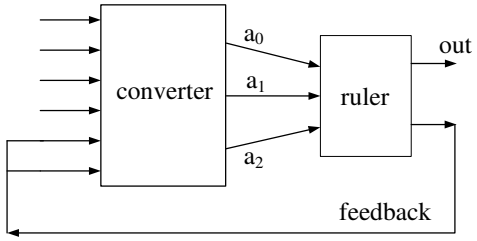

Fig. 9. Connect converter and neuron to assemble a more complex neuron.

2.2 Design of Arithmetic Circuits

In a different context the spiking neuron model has been proved to be more appropriate to simulate the Boolean logic than other traditional neuron models, such as threshold gates and sigma-pi units, [11].

In the area of SN P systems there have been several attempts to synthesize arithmetic operations with SN P neurons [5, 15]. In this paper we provide an hardware implementation for the basic HP and LP neurons and we sketch how using these two basic types any arithmetic circuit could be designed. In particular, we have designed an adder and a comparator as two relevant examples. The adder is the core component of any arithmetic logic unit. In today's digital computer, most of the arithmetic functions, such as multiplication, subtraction, and division, are constructed from a binary adder with the addition of other smaller circuits (e.g., inverter or shifter). However, there are two fundamental differences in an SN

P system from the traditional Boolean logic that must be considered when moving to a hardware implementation: (1) Number representation: In SN P systems, the input and output numbers are presented as natural numbers, instead of binary numbers. The exact encoding mechanism can be either the time interval between two spikes, or the number of continuous spikes, [15]. In this paper we use this last option. (2) Logic construction: A neuron of an SN P system (in short, an SN P neuron) follows different rules from Boolean logic gates and the network to realize a given arithmetic function is, in general, not directly extendable. As the SN P system design is still in its initial stage, the implemented SN P systems are not general-purpose but fully customized to the specific arithmetic functions.

For instance, a 1-bit full adder is presented in Figure 10.(a). It computes the sum of two natural numbers, with the maximum output value of 7. Note that the minimum number of SN P neurons needed for such a design is 4. Four additional neurons are adopted in order to improve the reliability of the adder, a strategy that will be explained in Section 4. A feedback is applied to handle the Carry-out (C_{out}) in the addition.

Figure 10.(b) shows the timing diagram of this adder. The delay from the primary input to the final output corresponds to the number of levels in the SN P system. In this specific design, there are three levels. Some neurons, such as the LP neuron in the second level, are introduced only as a buffer, in order to synchronize the operation across the different branches.

The other representative arithmetic circuit is the comparator that evaluates the absolute difference between two numbers and compares the value versus a given threshold. This function is widely used in many image-processing applications.

In a conventional digital design, the function of comparison is usually realized by a number of Boolean logic gates, such as XOR. Figure 11 presents a new design that is fully based on LP and HP neurons. For two input spiking trains, the comparator outputs 1 if the difference of two inputs is greater than or equal to 3. The SN P design is based on three parts: *Part 1:* Subtraction between the two inputs. *Part 2:* Comparison versus the threshold of 3. The threshold value determines the number of levels in the comparison network. Thus, if a different threshold value is needed, the chain of LP neurons in P_2 can (should) be proportionally adjusted while P_1 and P_3 remain the same. *Part 3:* Production of one or zero output spikes, depending on the comparison result.

In the design of an SN P system comparator, the maximum length of input spike trains is actually not limited since there is no feedback in the network (this is different from the previously presented design of an SN P system adder).

3 Error Model: The Loss of Spikes

In this paper we address the problem of engineering reliable computation circuits, based on SN P systems, using unreliable components. This is done by identifying what is generally called the "error model" of interest. In this section, we propose

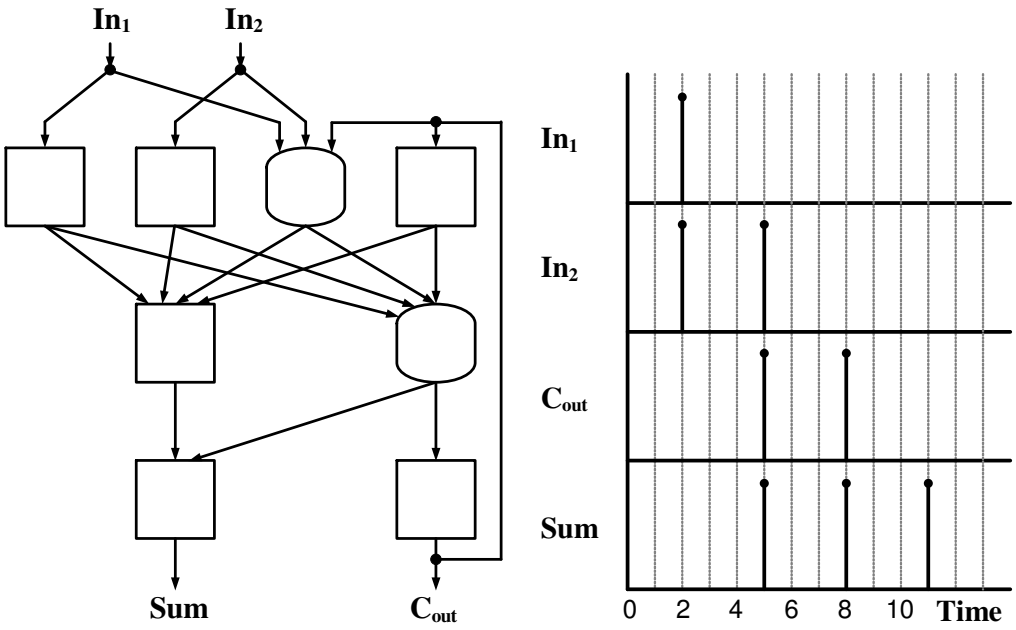

Fig. 10. Design of an SN P system working as 1-bit full adder and its time diagram.

a 1-bit error model to describe the error associated to a single SN P neuron. According to definition of firing, a neuron can only fire one spike at each time step. However, in a realistic scenario, it is reasonable to assume that an emitted spike is lost or, more generally, a spike does not reach a neuron at the designed time (for this reason, we call this problem as the "spike missing error"). For instance, this problem can happen when the spike arrives at the destination only after the neuron evaluates the inputs or the synapse fails to work.

For modeling and analysis feasibility, we evaluate the spike missing error considering the equivalent problem of shifting the threshold present in the HP and LP neurons. The threshold shifting error consists in the fact that the threshold in the firing rules shifts one more unit. The two types of problems are indeed equivalent, see Figure 12.

There is, however, a specific case in which these two types of error are not equivalent. In fact, when the number of input spikes is exactly one, the LP neuron with threshold equal to two will behave differently in the considered types of errors. The neuron with the threshold shifting error will fire while the neuron with spike missing will not fire because does not receive any input. However, if we assume that the spike missing problem is due to the fact that a single spike did not reach the target neuron before the neuron evaluates the other input spikes, we can realize that the spike missing error will never happen on the first spike and the situation

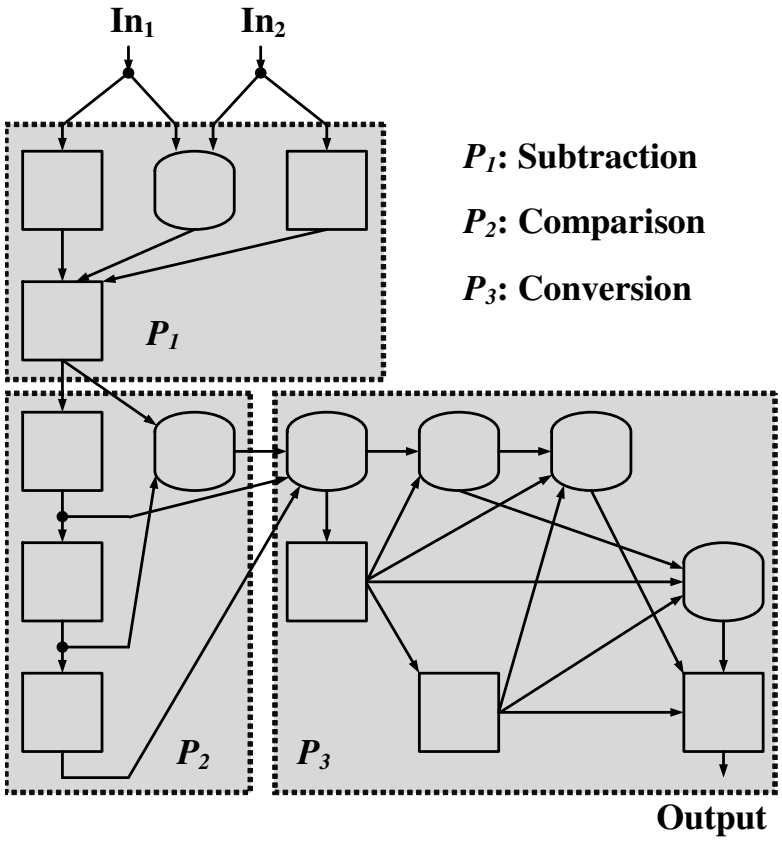

Fig. 11. Design of an SN P system comparator.

described cannot really occur (a neuron starts evaluating its input only after it receives its first spike).

From the above description, one can conclude that the one spike missing error is equivalent to a shift of the threshold by one in LP and HP neurons. Since the threshold shift error is simpler to analyze and model, we consider this as basic issue to study the reliability in SN P systems.

4 Reliability

4.1 Reliability Improvement of Single Neurons

In this section we analyze the reliability of single neurons and of arithmetic circuits by considering the error model previously proposed. Moreover we propose a design

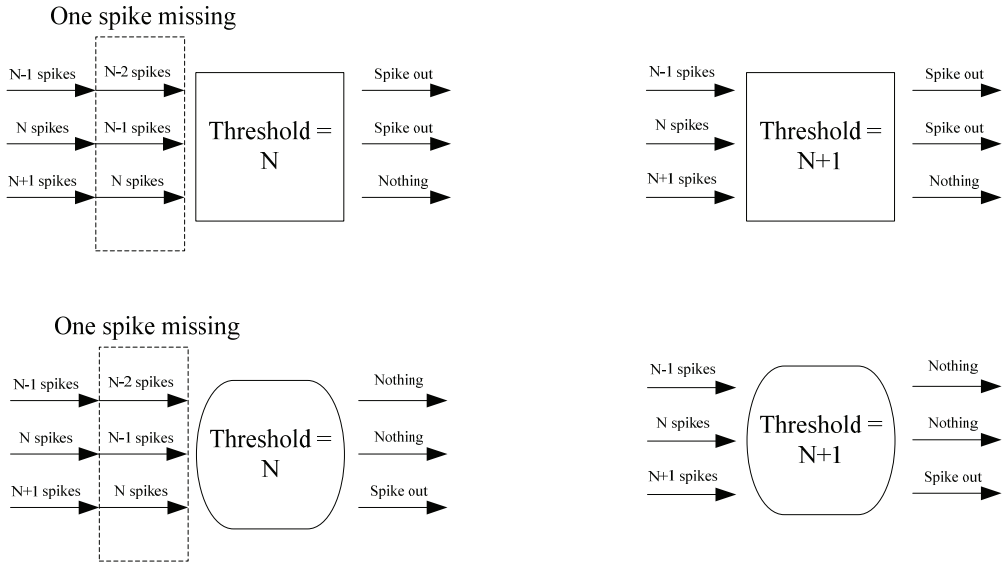

Fig. 12. Spike missing error and threshold shifting error for LP and HP neurons.

that can improve the reliability, which is more effective and efficient than the traditional Triple Modular Redundancy (TMR) (usually considered in the area). To systematically improve the error tolerance of an SN P system, our approach starts from enhancing the reliability of a single LP and HP neuron. For this purpose, a general solution is to add more identical units and a majority vote unit, such that the output will be correct if the majority of those multiple copies work correctly and the majority vote unit is correct. Figure 13.(a) illustrates an example of TMR for an LP neuron. Three copies of the LP neuron work in parallel; the HP neuron serves as the majority vote unit in this case. Based on the proposed error model, the TMR functions correctly when there is at most one erroneous neuron. If we assume that each neuron has an error rate of x then the success rate of the TMR circuit is $(1-x)^4 + 4x(1-x)^3$.

Although TMR is a general and widely used solution in many designs, a better solution should recognize the specific properties of the considered system and error model. Figure 13.(b) presents an alternative motif that is specifically designed for the LP neuron and the 1-bit error model. It uses the same number of neurons as TMR, but in a different topology. According to the considered error model, the error can only happen to the two inputs case. If the neuron has one input spike or three input spikes, the output is still correct. Thus avoiding two inputs cases is a good choice to improve reliability. The motif proposed in Figure 13.(b) fails only when HP neuron and the output LP neuron are erroneous. In this case the success rate is $1-x^2$. Figure 14 shows the success rate of TMR and the proposed motif as

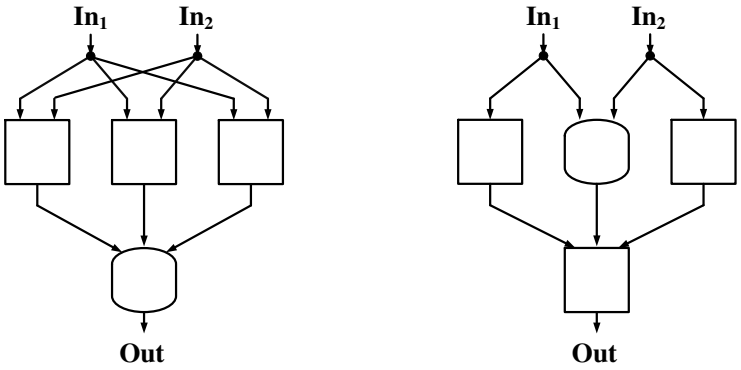

Fig. 13. Two methods to improve the reliability of a 2-input LP neuron. (a) TMR (b) Specific motif for an SN P system.

function of the error rate x. To achieve 95% success rate, the error rate that TMR can tolerate is 9.76% while the motif can tolerate up to 22.4%. Therefore, the proposed motif has much better reliability than TMR. In addition, the reliability of the motif can be further enhanced by adding more HP neurons in parallel with the one in Figure 13.(b). The success rate is presented in Figure 14 as well, which shows the asymptotic behavior. This suggests that such solution provides the flexibility to meet various reliability demands, at the cost of more neurons.

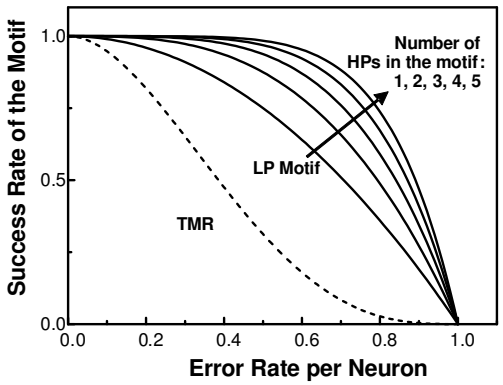

Fig. 14. The reliability of an LP neuron is significantly improved by the motif proposed in Figure 13.(b)

4.2 Reliability of Arithmetic Circuits

The motif described in Figure 13.(b) has been already used in the adder and comparator designs presented in Section 2 (Figure 10 and Figure 11). Figure 15 shows the reliability of the adder and comparator, versus the error rate of the neuron. The comparator has a lower reliability than the adder, due to its specific and more complex network structure.

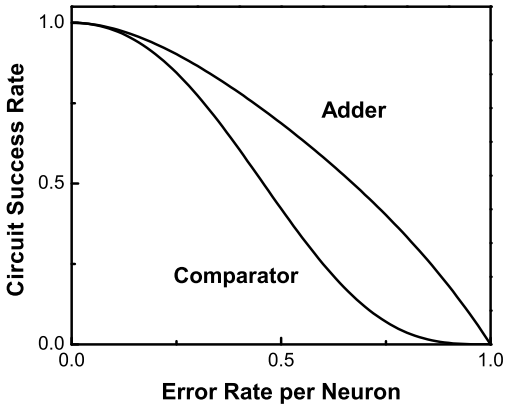

Fig. 15. Reliability of the SN P system adder and comparator.

Figure 16 presents a comparison between the SN P system adder and a CMOS-based adder. At the 95% success rate of the adder function, the SN P system design tolerates up to 17.2% error rate per neuron, as compared to 0.57% in the single module implementation or 4.4% error rate per NAND gate in the case of TMR.

As previously discussed, a general strategy to increase the reliability of SN P systems is to add more neurons. In this case it is important to evaluate the hardware cost versus the reliability as shown in Figure 17. The figure presents the error tolerance at 95% success rate of an adder in both cases, CMOS-based SN P system design (Section 5) and Boolean design. In the SN P system design, the tolerance of a single neuron error is rapidly enhanced by moderately addition of neurons. Overall we can then conclude that SN P systems are more robust than Boolean circuits and with lower area cost.

5 Implementation and Performance

Based on the designs presented in Section 2, this section investigates the effective hardware implementation that may benefits IC design as early as possible. CMOS

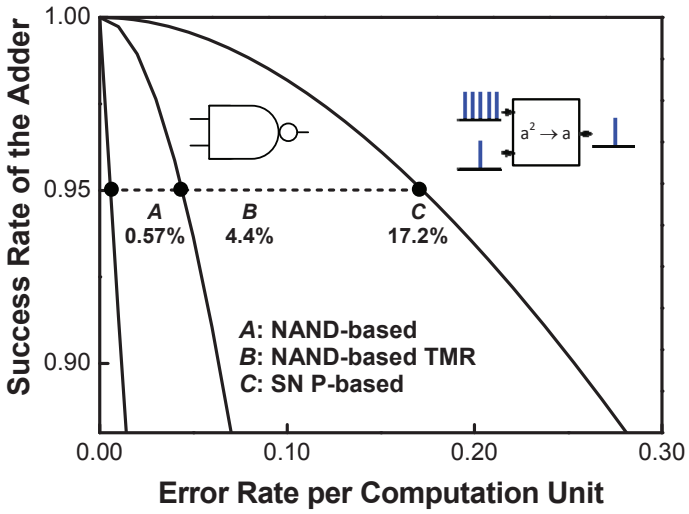

Fig. 16. The reliability of the SN P system adder is much better than the CMOS based adder.

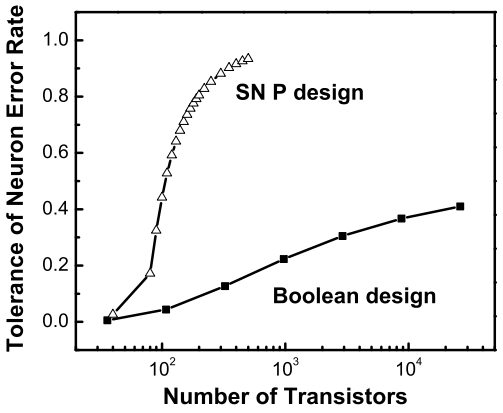

Fig. 17. The reliability of the SN P system is effectively improved with smaller area overhead than the Boolean design.

is selected for its technological maturity, although other emerging devices may offer a better potential [1, 12, 10]. $45nm$ PTM is used in all presented simulations, [16].

5.1 DRAM-type Neuron Design

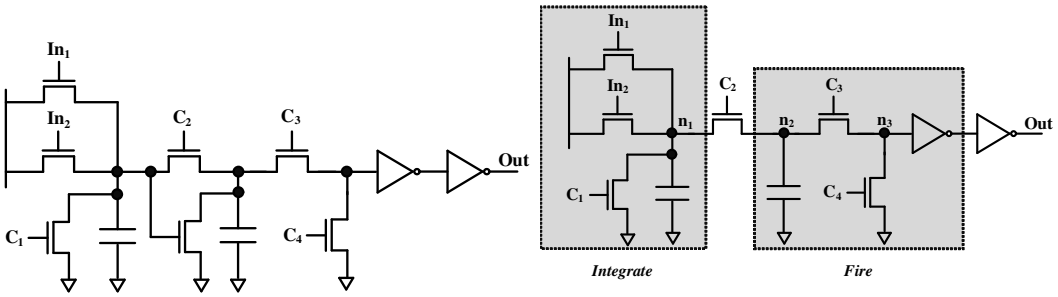

Fig. 18. CMOS-based neuron design, using DRAM-type structures.

Previous works in the neuromorphic field have proposed many analog and digital versions of CMOS circuits that emulate the realistic, yet complicated neuron behavior [6]. On the other side, this work targets the design of a simple circuit that will capture the essential features of SN P systems. A simple design is essential as it benefits the efficiency and the low power consumption. The exact designs for LP and HP neurons are presented in Figure 18, using 11 and 10 transistors, respectively. Their design is guided by the following two principles, one inspired from the hardware consideration, and the other one from the neurophysiological behavior of the neuron: (1) Memory-based computing: The operation of the neuron is better described as a finite state machine, rather than a logic function [11]. Thus, it is essential to have a memory to hold the current state and to decide the next state. This is akin to a lookup-table based computing.

(2) Leaky Integrate-Fire model (LIF): The LIF model is a classical one used to describe the biological process in a spiking neuron [9]. It contains three basic steps: the neuron integrates the input spikes, determines the number of output spikes, and leaks out the history. The presented neuron design follows these two principles. It adopts a DRAM-type structure to realize memory-based computing. Since the goal is to design arithmetic circuits, a long-term memory, such as Phase-Change-Memory or magnetic memory, [12, 10], may not be necessary; the leaky DRAM with short-term data storage serves well for our design purpose. As shown in Figure 18, the design can be viewed as three components, reproducing the LIF model: the first DRAM structure integrates all input spikes into a capacitor, on which the voltage level represents the integrated results. During the evaluation stage, the voltage is transferred to another smaller capacitor, which serves as the

output register; this voltage is then compared with the threshold voltage of an inverter to determine whether the firing or forgetting rule should be applied. After the firing/forgetting, two NMOS transistors, gated by C_1 and C_4, are turned on to reset those two capacitors; this operation emulates the leaky function. In addition, as all nodes in the neuron are reset to the ground level at the end of the operation cycle, the leakage power is minimized. Finally, a second inverter is added to restore the spike quality and to buffer the connection with other neurons. Figure 19 illustrates SPICE simulation results of a 2-input HP neuron. If there are more input channels, only the number of pass-gates in the integrate stage needs to the adjusted. The value of the capacitors is $10fF$, which can be realized from layout parasitic without pursuing explicit capacitors. Although four clock signals are applied to control different stages of the operation, they can be generated from a single clock source: for instance, C_1 and C_2 are delayed from C_3, and C_4 is the inverted version. Currently the neuron operates at 0.5 ns per cycle. This is limited by the RC constant during charging the capacitors, as well as the partition among various phases. The reset process does not limit the speed. Further optimization is feasible to improve the operation. Since both input and output signals are rail-to-rail voltage spikes, there is no issue to cascade neurons.

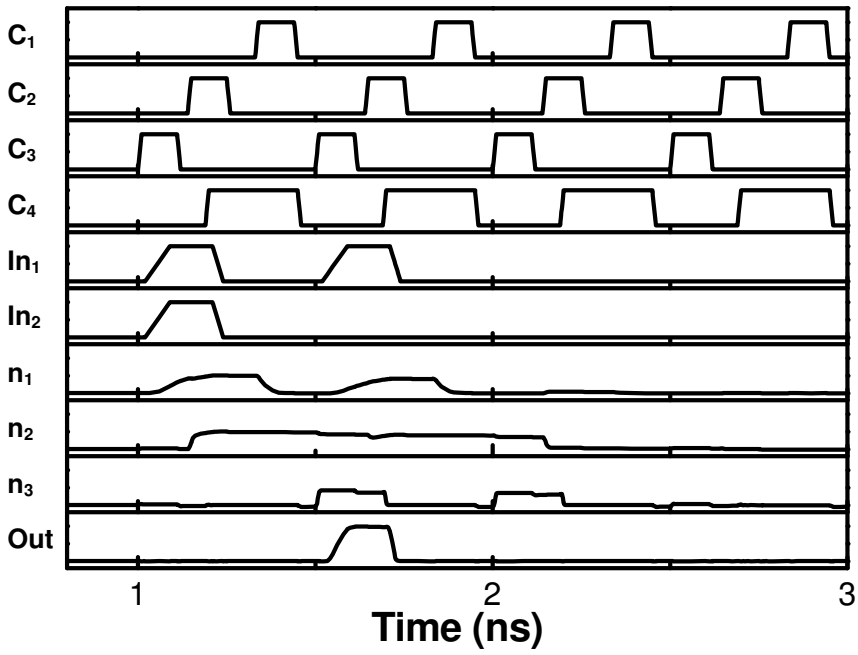

Fig. 19. SPICE simulation of a 2-input HP neuron.

As presented earlier, by connecting these neuron circuits into appropriate SN P systems, various functions can be realized. Figure 20 demonstrates SPICE simulations for the adder and the comparator, as described in Section 2. Different circuits can experience different signal delays that are proportional to the levels of the considered SN P system. The integrity of the spikes is, however, well maintained through the operation.

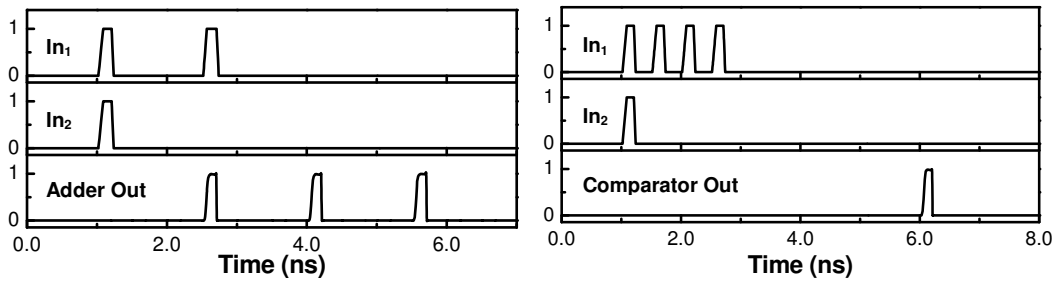

Fig. 20. The timing diagrams of SN P systems associated to arithmetic circuits.

5.2 Circuit Performance Benchmarks

As compared to other neuromorphic circuit design, [6], the proposed CMOS design is simple, implying the benefit in power and density, as well as the vulnerability under process variations. The pass-gate structure in the DRAM-type design may exacerbate this problem. On the other, by reducing the design cost at the neuron level with the price of reliability, the expectation is to better manage the global system reliability through the use of SN P system, as discussed in the previous sections. This section benchmarks major performance metrics, such as reliability, switching energy and the leakage. It uses $45nm$ SPICE simulation with threshold voltage (V_{th}) variations. Figure 21 samples the tolerance of (V_{th}) variance (σ_{Vth}) at single neuron and at circuit level. The circuits of LP and HP neurons have a different tolerance to (V_{th}) variance due to their specific circuit structures. Furthermore, if 95% success rate is used as criteria, it is observed that the tolerance to (V_{th}) variance is improved from a single neuron to the SN P system, confirming the reliability of the topology.

For time-free P system application, [2], it is important to evaluate the impact of stochastic delays for the application of spiking rules, [2]. Figure 22 presents the tolerance of stochastic delays (σ_{td}) of the adder and of the comparator ([2]). For 95% success rate, the circuits can tolerate up to $0.5ns$ clock cycle. A tradeoff can be made to improve the reliability versus signal delay by slowing down the clock cycle. Table 23 summarizes the evaluation of the energy consumption of SN P systems and of Boolean design based on the NAND gate. The switching energy

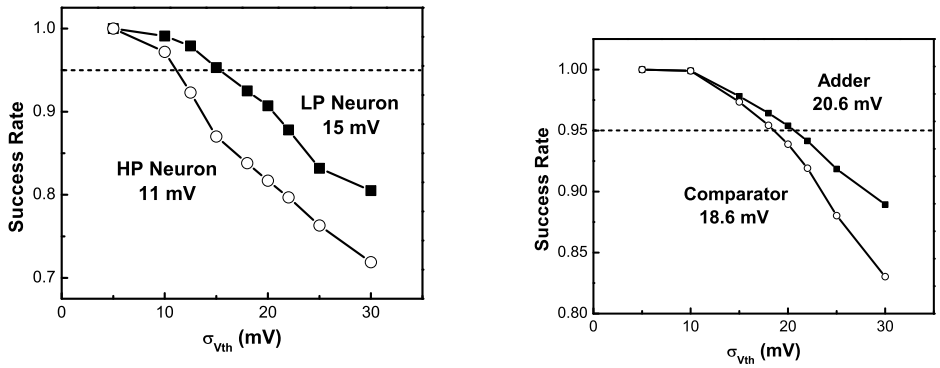

Fig. 21. SN P systems tolerate well (V_{th}) variations at single neuron and circuit levels.

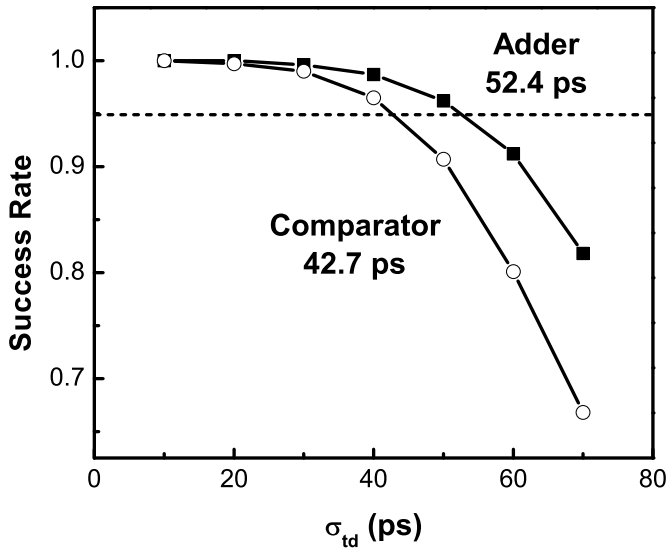

Fig. 22. SN P systems tolerate well stochastic delays.

is averaged over a set of input patterns that are shared between SN P systems and Boolean designs. For a simple circuit, such as the adder, the SN P system design costs more switching energy because its operation is more complicated. For instance, the SN P system neuron has to be reset at the end of every cycle. This condition is similar to that of a dynamic Boolean logic, which may cause more energy consumption in the next cycle. However, as circuit complexity goes up, the difference in switching energy goes down, partially because the size of an SN P system increases more slowly than that of Boolean circuits. SN P systems have

Function	Design Style	Switching Energy (fJ)	Leakage Power (μW)
Adder	Boolean	5.76	2.15
	SN P	83.8	0.55
Comparator	Boolean	83.73	38.52
	SN P	151.8	0.91

Fig. 23. The benchmark of energy and leakage.

much lower leakage power, benefiting from the fact that is reset at each cycle. Overall, the design style of SN P systems may reflect different tradeoffs between active and leakage power.

6 Discussion

This work represents the first step toward the IC design based on the principles and concepts of SN P systems. The proposed design reveals promising features such as the feasibility of large-scale integration, superior to the one obtained in the conventional Boolean logic. In this paper we have shown the implementation of simple SN P system neurons using DRAM-type CMOS circuits. We have provided a quantitative analysis by using $45nm$ simulations that illustrate how the operation of a single SN P neuron and of an assembled SN P system shows superior reliability at the circuit level and tolerate a larger amount of unreliability, than the one obtained in the Boolean design. These results seems very promising and we expect more future results on the hardware implementation of SN P systems, on the design optimization of SN P neurons and on the integration with emerging nanoelectronics.

References

1. Bernstein K., Cavin R.K., Porod W., Seabaugh A., Welser J. Device and Architecture Outlook for Beyond CMOS Switches. *Proc. IEEE*, 98, 12, 2010.

2. Cavaliere M., Mura I. Experiments on the Reliability of Stochastic Spiking Neural P Systems. *Natural Computing* 7, 4, 2008.
3. Cavaliere M., Sburlan D. Time-independent P systems. In *Membrane Computing. International Workshop WMC5, Milano, Italy, 2004*, LNCS 3365, Springer, 2005, pp. 239–258.
4. Gerstner W.. Population Dynamics of Spiking Neurons: Fast Transients, Asynchronous States, and Locking. *Neural Computation*, 12, 43, 2000.
5. Guitérrez-Naranjo M.A., Leporati A. First Steps towards a CPU made of Spiking Neural P Systems. *J. Comput. Commun. Control*, 5, 3, 2009.
6. Indiveri G. et al. Neuromoprhic Silicon Neuron Circuits. *Frontiers in Neuroscience*, 5, 73, 2011.
7. Păun Gh. Spiking Neural P Systems: A Tutorial. *Bulletin of the EATCS*, 91 (Feb 2007).
8. Păun Gh., Rozenberg G., Salomaa A. Eds. *The Oxford Handbook of Membrane Computing.* Oxford University Press, 2010.
9. Knight B.W. Dynamics of Encoding in a Population of Neurons. *J. Gen. Physiol.*, 59, 6, 1972.
10. Kuzum D., Jeyasingh R.G.D., Lee B., Wong H.-S.P. Nanoelectronic Programmable Synapses based on Phase Change Materials for Brain-Inspired Computing. *Nano Lett*, 12, 5, 2011.
11. Schmitt M. On Computing Boolean Functions by a Spiking Neuron. *J. Annals of Mathematics and Artificial Intelligence*, 24, 1-4, 1998.
12. Sharad M., Augustine C., Panagopoulos G., Roy K. Spin-Based Neuron Model with Domain-Wall Magnets as Synapse. *Trans. Nanotechnology*, 11, 4, 2012.
13. Song T., Maciás-Ramos L.F., Pan L., Peŕez-Jiménez M.J. Time-Free Solution to SAT Problem Using P Systems with Active Membranes. *Theoretical Computer Science*, 529, 2014.
14. Pan L., Zeng X., Zhang X. Time-Free Spiking Neural P Systems. *Neural Computation*, 23, 5, 2011.
15. Zeng X., Song T., Zhang X., Pan L. Performing four Basic Arithmetic Operations with Spiking Neural P Systems. *Trans. NanoBioscience*, 11, 4, 2012.
16. Predictive Technology Model, available at http://ptm.asu.edu